OpenSMILES specification

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
1.0	2012-29-09	Current specification	CAJ

Contents

www.opensmiles.org

Contributors: Richard Apodaca, Noel O'Boyle, Andrew Dalke, John van Drie, Peter Ertl, Geoff Hutchison, Craig A. James, Greg Landrum, Chris Morley, Egon Willighagen, Hans De Winter, Tim Vandermeersch

1 Introduction

> "... we cannot improve the language of any science, without, at the same time improving the science itself; neither can we, on the other hand, improve a science, without improving the language or nomenclature which belongs to it ... "
>
> Antoine Lavoiser, 1787

1.1 Purpose

This document formally defines an open specification version of the SMILES language, a typographical line notation for specifying chemical structure. It is hosted under the banner of the Blue Obelisk project, with the intent to solicit contributions and comments from the entire computational chemistry community.

1.2 Motivation

SMILES was originally developed as a proprietary specification by Daylight Chemical Information Systems Since the introduction of SMILES in the late 1980's, it has become widely accepted as a defacto standard for exchange of molecular structures. Many independent SMILES software packages have been written in `C`, `C++`, `Java`, `Python`, `LISP`, and probably even `FORTRAN`.

At this point in the history of SMILES, is appropriate for the chemistry community to develop a new, non-proprietary specification for the SMILES language. Daylight's SMILES Theory Manual has long been the "gold standard" for the SMILES language, but as a proprietary specification, it limits the universal adoption of SMILES, and has no mechanism for contributions from the chemistry community. We salute Daylight for their past contributions, and the excellent SMILES documentation they provided free of charge for the past two decades.

1.3 Audience

This document is intended for developers designing or improving a SMILES parser or writer. **Readers are expected to be acquainted with SMILES.** Due to the formality of this document, it is not a good tutorial for those trying to learn SMILES. This document is written with **precision** as the primary goal; **readability** is secondary.

1.4 What is a Molecule? The Valence Model of Chemistry

Before defining the SMILES language, it is important to state the physical model on which it is based: the valence model of chemistry, which uses a mathematician's graph to represent a molecule. In a chemical graph, the **nodes** are atoms, and the **edges** are semi-rigid bonds that can be single, double, or triple according to the rules of valence bond theory.

This simple mental model has little resemblance to the underlying quantum-mechanical reality of electrons, protons and neutrons, yet it has proved to be a remarkably useful approximation of how atoms behave in close proximity to one another. However, the valence model is an imperfect representation of molecular structure, and the SMILES language inherits these imperfections. Chemical bonds are often tautomeric, aromatic or otherwise fractional rather than neat integer multiples. Delocalized bonds, bond-centered bonds, hydrogen bonds and various other inter-atom forces that are well characterized by a quantum-mechanics description simply don't fit into the valence model.

> "If you can build a molecule from a modeling kit, you can name it."
>
> McLeod Peters

McLeod and Peter's quip captures the deficiencies of SMILES well: if you **can't** build a molecule from a modeling kit, the deficiencies of SMILES and other connection-table formats become apparent.

2 Formal Grammar

2.1 Syntax versus Semantics

This SMILES specification is divided into two distinct parts: A **syntactic specification** specifies how the atoms, bonds, parentheses, digits and so forth are represented, and a **semantic specification** that describes how those symbols are interpreted as a sensible molecule. For example, the syntax specifies how ring closures are written, but the semantics require that they come in pairs. Likewise, the syntax specifies how atomic elements are written, but the semantics determines whether a particular ring system is actually aromatic.

For this specification, the syntax and semantics are explained separately; in practice, the syntax and semantics are usually mixed together in the code that implements a SMILES parser. This chapter is only concerned with syntax.

2.2 Grammar

Section	**Formal Grammar**
ATOMS	
Atoms	*atom* ::= *bracket_atom* | *aliphatic_organic* | *aromatic_organic* | '*'
ORGANIC SUBSET ATOMS	
Organic Subset	*aliphatic_organic* ::= 'B' | 'C' | 'N' | 'O' | 'S' | 'P' | 'F' | 'Cl' | 'Br' | 'I'
	aromatic_organic ::= 'b' | 'c' | 'n' | 'o' | 's' | 'p'
BRACKET ATOMS	
Bracket Atoms	*bracket_atom* ::= '[' *isotope*? *symbol* *chiral*? *hcount*? *charge*? *class*? ']'
	symbol ::= *element_symbols* | *aromatic_symbols* | '*'
	isotope ::= *NUMBER*
	element_symbols ::= 'H' | 'He' | 'Li' | 'Be' | 'B' | 'C' | 'N' | 'O' | 'F' | 'Ne' | 'Na' | 'Mg' | 'Al' | 'Si' | 'P' | 'S' | 'Cl' | 'Ar' | 'K' | 'Ca' | 'Sc' | 'Ti' | 'V' | 'Cr' | 'Mn' | 'Fe' | 'Co' | 'Ni' | 'Cu' | 'Zn' | 'Ga' | 'Ge' | 'As' | 'Se' | 'Br' | 'Kr' | 'Rb' | 'Sr' | 'Y' | 'Zr' | 'Nb' | 'Mo' | 'Tc' | 'Ru' | 'Rh' | 'Pd' | 'Ag' | 'Cd' | 'In' | 'Sn' | 'Sb' | 'Te' | 'I' | 'Xe' | 'Cs' | 'Ba' | 'Hf' | 'Ta' | 'W' | 'Re' | 'Os' | 'Ir' | 'Pt' | 'Au' | 'Hg' | 'Tl' | 'Pb' | 'Bi' | 'Po' | 'At' | 'Rn' | 'Fr' | 'Ra' | 'Rf' | 'Db' | 'Sg' | 'Bh' | 'Hs' | 'Mt' | 'Ds' | 'Rg' | 'Cn' | 'Fl' | 'Lv' | 'La' | 'Ce' | 'Pr' | 'Nd' | 'Pm' | 'Sm' | 'Eu' | 'Gd' | 'Tb' | 'Dy' | 'Ho' | 'Er' | 'Tm' | 'Yb' | 'Lu' | 'Ac' | 'Th' | 'Pa' | 'U' | 'Np' | 'Pu' | 'Am' | 'Cm' | 'Bk' | 'Cf' | 'Es' | 'Fm' | 'Md' | 'No' | 'Lr'
	aromatic_symbols ::= 'b' | 'c' | 'n' | 'o' | 'p' | 's' | 'se' | 'as'
CHIRALITY	
Chirality	*chiral* ::= '@' | '@@' | '\@TH1' | '\@TH2' | '\@AL1' | '\@AL2' | '\@SP1' | '\@SP2' | '\@SP3' | '\@TB1' | '\@TB2' | '\@TB3' | ... | '\@TB20' | '\@OH1' | '\@OH2' | '\@OH3' | ... | '\@OH30'
HYDROGENS	
Hydrogens	*hcount* ::= 'H' | 'H' *DIGIT*
CHARGES	
Charge	*charge* ::= '-' | '-' *DIGIT*? *DIGIT* | '+' | '+' *DIGIT*? *DIGIT* | '--' **deprecated** | '++' **deprecated**
ATOM CLASS	
Atom Class	*class* ::= ':' *NUMBER*

Section	Formal Grammar
BONDS AND CHAINS	
Bonds	bond ::= '-' | '=' | '#' | '$' | ':' | '/' | '\'
	ringbond ::= bond? DIGIT | bond? '%' DIGIT DIGIT
	branched_atom ::= atom ringbond* branch*
	branch ::= '(' chain ')' | '(' bond chain ')' | '(' dot chain ')'
	chain ::= branched_atom | chain branched_atom | chain bond branched_atom | chain dot branched_atom
	dot ::= '.'
SMILES STRINGS	
	smiles ::= terminator | chain terminator
	terminator ::= SPACE | TAB | LINEFEED | CARRIAGE_RETURN | END_OF_STRING

3 Reading SMILES

3.1 Atoms

3.1.1 Atomic Symbol

An atom is represented by its atomic symbol, enclosed in square brackets, `[]`. The first character of the symbol is uppercase and the second (if any) is lowercase, except that for aromatic atoms (see below), the first character is lowercase. There are 114 valid atomic symbols , as defined by IUPAC (see also Web Elements).

The symbol `'*'` is also accepted as a valid atomic symbol, and represents a "wildcard" or unknown atom.

Examples:

SMILES	Atomic Symbol
`[U]`	Uranium
`[Pb]`	Lead
`[He]`	Helium
`[*]`	Unkwown atom

3.1.2 Hydrogens

Hydrogens inside of brackets are specified as `Hn` where `n` is a number such as `H3`. If no `Hn` is specified, it is identical to `H0`. If `H` is specified without a number, it is identical to `H1`. For example, `[C]` and `[CH0]` are identical, and `[CH]` and `[CH1]` are identical.

Hydrogens that are specified in brackets with this notation have undefined isotope, no chirality, no other bound hydrogen, neutral charge, and an undefined atom class.

Examples:

SMILES	Name	Comments
`[CH4]`	methane	
`[ClH]`	hydrochloric acid	`H1` implied
`[ClH1]`	hydrochloric acid	

A hydrogen atom cannot have a hydrogen count, for example, `[HH1]` is illegal. Hydrogens connected to other hydrogens must be represented as explicit atoms in square brackets. For example molecular hydrogen must be written as `[H][H]`.

Question: are more than 9 hydrogens possible? Should they be supported?

3.1.3 Charge

Charge is specified by a `+n` or `-n` where `n` is a number; if the number is missing, it means either `+1` or `-1` as appropriate.

For backwards compatibility, a general-purpose SMILES parser should accept the symbols `--` and `++` to mean charges of `-2` and `+2`, but this is a deprecated form and should be avoided.

Examples:

SMILES	Name	Comments
`[Cl-]`	chloride anion	`-1` charge, `H0` implied
`[OH1-]`	hydroxyl anion	`-1` charge, `H1`
`[OH-1]`	hydroxyl anion	`-1` charge, `H1`
`[Cu+2]`	coppor cation	`+2` charge, `H0` implied
`[Cu++]`	copper anion	`+2` charge, `H0` implied

An implementation is required to accept charges in the range `-15` to `+15`.

3.1.4 Isotopes

Isotopic specification is placed inside the square brackets for an atom preceeding the atomic symbol; for example:

SMILES	Atomic Symbol
`[13CH4]`	methane
`[2H+]`	deuterium ion
`[238U]`	Uranium 238 atom

An isotope is interpreted as a number, so that `[2H]`, `[02H]` and `[002H]` all mean deuterium. If the isotope field is not specified then the atom is assumed to have the naturally-occuring isotopic ratios. The isotope value 0 also indicates an isotope of zero, that is `[0S]` is **not** the same as `[S]`.

There is no requirement that the isotope is a genuine isotope of the element. Thus, `[36Cl]` is allowed even though ^{35}Cl and ^{37}Cl are the actual known stable isotopes of chlorine.

A general-purpose SMILES parser must accept at least three digits for the isotope and values from 0 to 999.

3.1.5 Organic Subset

A special subset of elements called the "organic subset" of **B**, **C**, **N**, **O**, **P**, **S**, **F**, **Cl**, **Br**, **I**, and ***** (the "wildcard" atom) can be written using the only the atomic symbol (that is, without the square brackets, H-count, etc.). An atom is specified this way has the following properties:

- "implicit hydrogens" are added such that valence of the atom is in the lowest normal state for that element
- the atom's charge is zero
- the atom has no isotopic specification
- the atom has no chiral specification

The implicit hydrogen count is determined by summing the bond orders of the bonds connected to the atom. If that sum is equal to a known valence for the element or is greater than any known valence then the implicit hydrogen count is 0. Otherwise the implicit hydrogen count is the difference between that sum and the next highest known valence.

The "normal valence" for these elements is defined as:

Element	Valence
B	3

Element	Valence
C	4
N	3 or 5
O	2
P	3 or 5
S	2, 4 or 6
halogens	1
*	unspecified

Examples:

SMILES	Name
`C`	methane
`N`	ammonia
`Cl`	hydrochloric acid

Note: The remaining atom properties, chirality and ring-closures, are discussed in later sections.

3.1.6 The Wildcard `'*'` Atomic Symbol

The `'*'` atom represents an atom whose atomic number is unknown or unspecified. If it occurs inside brackets, it can have its isotope, chirality, hydrogen count and charge specified. If it occurs outside of brackets, it has no assumed isotope, a mass of zero, unspecified chirality, a hydrogen count of zero, and a charge of zero.

SMILES	Name
`Oc1c(*)cccc1`	ortho-substituted phenol

The `'*'` atom does not have any specific electronic properties or valence. If specified outside of square brackets, it takes on the valence implied by its bonds. If it is inside square brackets, it takes on the valence implied by its bonds, hydrogens and/or charge.

A `'*'` atom can be part of an aromatic ring. When deducing the aromaticity of a ring system, the ring system is considered aromatic if there is an element which could replace the `'*'` and make the ring system meet the aromaticity rules (see Aromaticity, below).

3.1.7 Atom Class

An "atom class" is an arbitrary integer, a number that has no chemical meaning. It is used by applications to mark atoms in ways that are meaningful only to the application. Multiple atoms may be labeled with the same atom class.

The atom class is specified after all other properties in square brackets. For example:

SMILES	Name
`[CH4:2]`	methane, atom's class is 2

If the atom class is not specified then the atom class is zero. The atom class is interpreted as a number, so both `[CH2:5]` and `[NH4+:005]` have an atom class of 5.

3.2 Bonds

Atoms that are adjacent in a SMILES string are assumed to be joined by a single or aromatic bond (see Aromaticity). For example:

SMILES	Name
CC	ethane
CCO	ethanol
NCCCC	n-butylamine
CCCCN	n-butylamine

Double, triple and quadruple bonds are represented by '=', '#', and '$' respectively:

SMILES	Name
C=C	ethane
C#N	hydrogen cyanide
CC#CC	2-butyne
CCC=O	propanol
[Rh-](Cl)(Cl)(Cl)(Cl)$[Rh-](Cl)(Cl)(Cl)Cl	octachlorodirhenate (III)

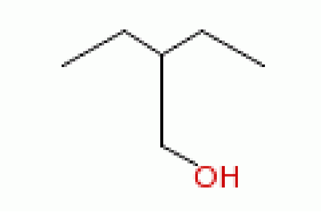
A single bond can be explicitely represented with '-', but it is rarely necessary.

SMILES	
C-C	same as: CC
C-C-O	same as: CCO
C-C=C-C	same as: CC=CC

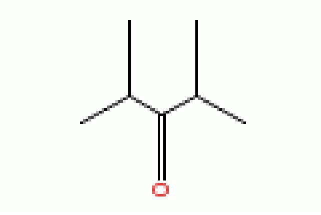
Note: The remaining bond symbols, ':\/', are discussed in later sections.

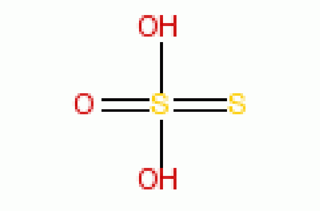
3.3 Branches

An atom with three or more bonds is called a **branched atom**, and is represented using parentheses.

Depiction	SMILES	Name
	CCC(CC)CO	2-ethyl-1-butanol

Branches can be nested or "stacked" to any depth:

Depiction	SMILES	Name
	CC(C)C(=O)C(C)C	2,4-dimethyl-3-penthanone
pic here	OCC(CCC)C(C(C)C)CCC	2-propyl-3-isopropyl-1-propanol

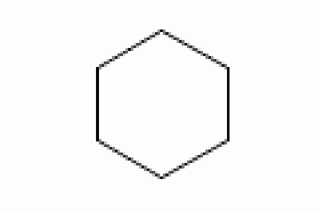
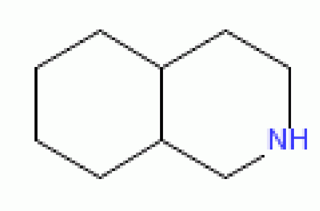
Depiction	SMILES	Name
	`OS(=O)(=S)O`	thiosulfate

The SMILES branch/chain rules allow nested parenthetical expressions (branches) to an arbitrary depth. For example, the following SMILES, though peculiar, is legal:

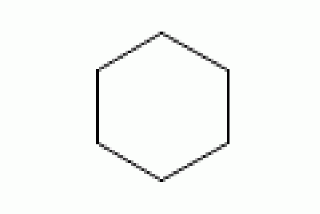
SMILES	Formula
`C(C(C(C(C(C(C(C(C(C(C(C(C(C(C(C(C(C(C(C(C))))))))))))))))))))C`	C22H46

3.4 Rings

In a SMILES string such as "C1CCCCC1", the first occurance of a ring-closure number (an "rnum") creates an "open bond" to the atom that preceeds the ring-closure number (the "rnum"). When that same rnum is encountered later in the string, a bond is made between the two atoms, which typically forms a cyclic structure.

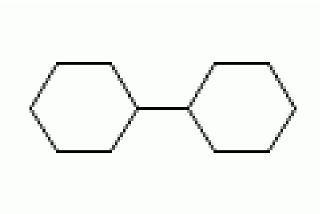
Depiction	SMILES	Name
	`C1CCCCC1`	cyclohexane
	`N1CC2CCCC2CC1`	perhydroisoquinoline

If a bond symbol is present between the atom and rnum, it can be present on **either or both** bonded atoms. However, if it appears on both bonded atoms, the two bond symbols must be the same.

Depiction	SMILES	Name
	`C=1CCCCC=1`	cyclohexene
	`C=1CCCCC1`	cyclohexene **(preferred from)**
	`C1CCCCC=1`	cyclohexene
	`C-1CCCCC=1`	**invalid**

Ring closures must be matched pairs in a SMILES string, for example, `C1CCC` is not a valid SMILES.

It is permissable to re-use ring-closure numbers. Once a particular number has been encountered twice, that number is available again for subsequent ring closures.

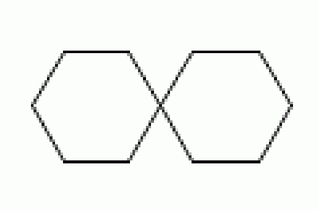
Depiction	SMILES	Name	Comment
	C1CCCCC1C1CCCCC1	dicyclohexyl	both SMILES are valid
	C1CCCCC1C2CCCCC2	dicyclohexyl	

Note that the ring number zero is valid, for example cyclohexane can be written C0CCCCC0.

Two-digit ring numbers are permitted, but must be preceeded by the percent '%' symbol, such as C%25CCCCC%25 for cyclohexane. Three-digit numbers and larger are never permitted. However, note that three digits are not invalid; for example, C%123 is the same as C3%12, that is, an atom with two rnum specifications.

The digit(s) representing a ring-closure are interpreted as a number, not a symbol, and two rnums match if their numbers match. Thus, C1CCCCC%01 is a valid SMILES and is the same as C1CCCCC1. Likewise, C%00CCCCC%00 is a valid SMILES.

A single atom can have several ring-closure numbers, such as this spiro atom:

Depiction	SMILES	Name
	C12(CCCCC1)CCCCC2	spiro[5.5]undecane

Two atoms cannot be joined by more than one bond, and an atom cannot be bonded to itself. For example, the following are not allowed:

SMILES	Comments
C12CCCCC12	**illegal, two bonds between one pair of atoms**
C12C2CCC1	**illegal, two bonds between one pair of atoms**
C11	**illegal, atom bonded to itself**

3.5 Aromaticity

3.5.1 The Meaning of "Aromaticity" in SMILES

"Aromaticity" in SMILES is primarily for cheminformatics purposes. In a cheminformatics system, we'd like to have a single representation for each molecule. The Kekule form masks the inherent uniformity of the bonds in an aromatic ring. SMILES uses a simplified definition of aromaticity that facilitates substructure and exact-structure searches, as well as Normalization and Canonicalization of SMILES.

The definition of "aromaticity" in SMILES is **not** intended to imply anything about the physical or chemical properties of a substance. In many or most cases, the SMILES definition of aromaticity will match the chemist's notion of what is aromatic, but in some cases it will not.

3.5.2 Kekule and Aromatic Representations

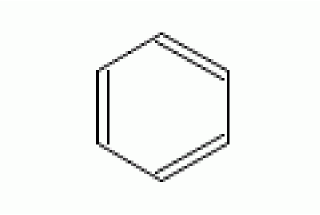
Aromaticity can be represented in one of two ways in a SMILES.

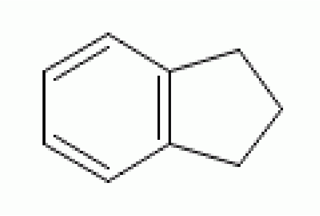
- In the Kekule form, using alternating single and double bonds, with uppercase symbols for the atoms.

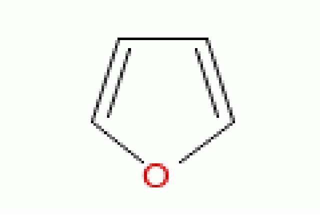
- An atomic symbol that begins with a lowercase letter is an aromatic atom, such as `'c'` for aromatic carbon. When aromatic symbols are used, no bond symbols are needed.

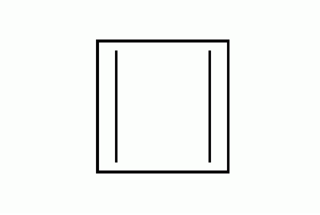
A lowercase aromatic symbol is defined as an atom in the sp^2 configuration in an aromatic or anti-aromatic ring system. For example:

Depiction	SMILES	Name

	`c1ccccc1`
	`C1=CC=CC=C1`
		benzene

	`c1ccc2CCCc2c1`
	`C1=CC=CC(CCC2)=C12`
		indane

	`c1occc1`
	`C1O=CC=C1`
		furan

	`c1ccc1`
	`C1=CC=C1`
		cyclobutadiene

The Kekule form is always acceptable for SMILES input. For output, the aromatic form (using lowercase letters) is preferred. The lowercase symbols eliminate the arbitrary choice of how to assign the single and double bonds, and provide a normalized form that more accurately reflects the electronic configuration.

3.5.3 Extended Hueckel's Rule

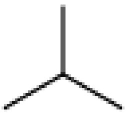
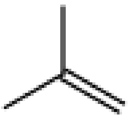
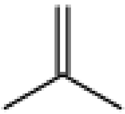
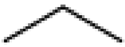
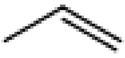
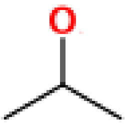
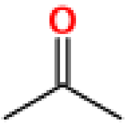
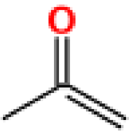
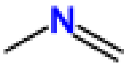
THIS SECTION IS UNDER MAJOR REVISION, AND AT THIS POINT IS ONLY FOR DISCUSSION PURPOSES.

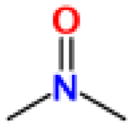
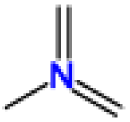
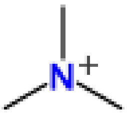
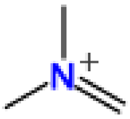
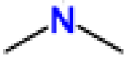
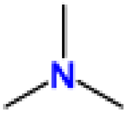
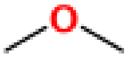
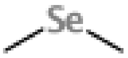
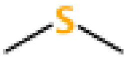
This proposed section is an attempt to simplify the rule-based system by enumerating all atom/bond configurations that are known to participate in aromatic systems.

A single, isolated ring that meets the following criteria is aromatic:

- All atoms must be sp^2 hybridized.
- The number of availabile "shared" π electrons must equal 4N+2 where N = 1, 2 or 3 (Huckel's rule).

Each element that can participate in an aromatic ring is defined to have the following number of π electrons:

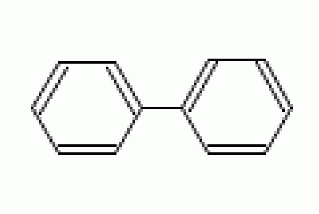
Configuration	π Electrons	Example
	1	`c1ccccc1`
	1	`c1ccccc1`
	1	`c1ccccc1`
	1	`c1ccccc1`
	1	`c1ccccc1`
	0-1	`c1ccccc1`
	0	`c1ccccc1`
	1	`c1ccccc1`
	1	`c1ccccc1`

Configuration	π Electrons	Example
	1	c1ccccc1
	1	c1ccccc1
	1	c1ccccc1
	1	c1ccccc1
	1-2	c1ccccc1
	1	c1ccccc1
	2	c1ccccc1
	2	c1ccccc1
	2	c1ccccc1

3.5.4 Aromaticity Algorithm

In an aromatic system, all of the aromatic atoms must be sp:sup:2 hybridized, and the number of π electrons must meet Huckel's 4n+2 criterion When parsing a SMILES, a parser must note the aromatic designation of each atom on input, then when the parsing is complete, the SMILES software must verify that electrons can be assigned without violating the valence rules, consistent with the sp^2 markings, the specified or implied hydrogens, external bonds, and charges on the atoms.

The aromatic-bond symbol `':'` can be used between aromatic atoms, but it is never necessary; a bond between two aromatic atoms is assumed to be aromatic unless it is explicitely represented as a single bond `'-'`. However, a single bond (nonaromatic bond) between two aromatic atoms **must** be explicitely represented. For example:

Depiction	SMILES	Name
	`c1ccccc1-c2ccccc2`	biphenyl

Note: Some SMILES parsers interpret a lowercase letter as sp^2 anywhere it appears; for example, `CccccC` would be interpreted as `CC=CC=CC`. The OpenSMILES specification does not allow this interpretation unless nonstandard parsing is explicitely allowed by the user.

3.6 More about Hydrogen

Hydrogens in a SMILES can be represented in three different ways:

Method	SMILES	Name	Comments
implicit hydrogen	`C`	methane	h-count deduced from normal valence (4)
atom property	`[CH4]`	methane	h-count specified for heavy atom
explicit hydrogen	`[H]C([H])([H])[H]`	methane	hydrogens represented as normal atoms

All three forms are equivalent. However, some situations require that one form must be used:

- Implicit hydrogen count may only be used for elements of the organic elements subset.
- Any atom that is specified with square brackets **must** have its attached hydrogens explicitely represented, either as a hydrogen count or as normal atoms.

A hydrogen that meets one of the following criteria must be represented as an explicit atom:

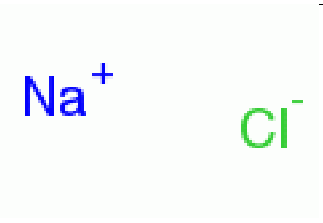
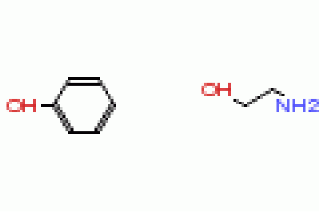
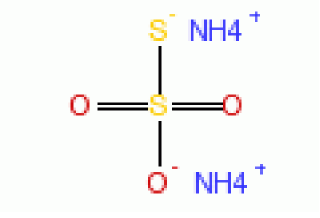
- hydrogens with charge (`[H+]`)
- a hydrogen connected to another hydrogen (such as molecular hydrogen, `[H][H]`)
- hydrogens with more than one bond (bridging hydrogens)
- Deuterium `[2H]` and tritium `[3H]`

It is permissible to use a mixture of an atom h-count and explicit hydrogen. In such a case, the atom's hydrogen count is the sum of the atomic h-count property and the number of attached hydrogens. For example:

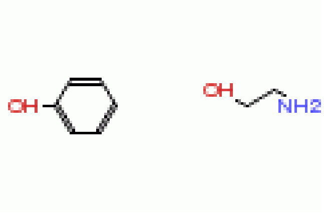
SMILES	Name
`[CH4]`	methane
`[H][CH2][H]`	methane
`[2H][CH2]C`	deuteroethane

3.7 Disconnected Structures

The dot `'.'` symbol (also called a "dot bond") is legal most places where a bond symbol would occur, but indicates that the atoms are **not** bonded. The most common use of the dot-bond symbol is to represent disconnected and ionic compounds.

Depiction	SMILES	Name
	`[Na+].[Cl-]`	sodium chloride
	`Oc1ccccc1.NCCO`	phenol, 2-amino ethanol
	`[NH4+].[NH4+].[O-]S(=O)(=O)[S-]`	diammonium thiosulfate

The dot can appear most places that a bond symbol is allowed, for example, the phenol example above can also be written:

Depiction	SMILES	Name
	`c1cc(O.NCCO)ccc1`	phenol, 2-amino ethanol
	`Oc1cc(.NCCO)ccc1`	phenal, 2-amino ethanol

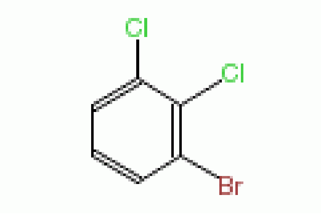
The second example above is an odd, but legal, use of parentheses and the dot bond, since the syntax allows a dot most places a regular bond could appear (the exception is that a dot can't appear before a ring-closure digit).

Although dot-bonds are commonly used to represent compounds with disconnected parts, a dot-bond does **not** in itself mean that there are disconnected parts in the compound. See the following section regarding ring digits for some examples that illustrate this.

The dot bond cannot be used in front of a ring-closure digit. For example, `C.1CCCCC.1` is illegal.

3.7.1 Other Uses of Ring Numbers and Dot Bond

A ring-number specifications ("rnum") is most commonly used to specify a ring-closure bond, but when used with the `'.'` dot-bond symbol, it can also specify a non-ring bond. Two rnums in a SMILES mean that the two atoms that preceed the rnums are bonded. A dot-bond `'.'` means that the atoms to which it is adjacent in the SMILES string are **not** bonded to each other. By combining these two constructs, one can "piece together" fragments of SMILES into a whole molecule. The following SMILES illustrate this:

SMILES/Depiction	Fragment SMILES	Name
`CC`	`C1.C1`	ethane
`CCC`	`C1.C12.C2`	propane
	`c1c2c3c4cc1.Br2.Cl3.Cl4`	1-bromo-2,3-dichlorobenzene

This feature of SMILES provides a convenient method of enumerating the molecules of a combinatorial library using string concatenation.

3.8 Stereochemistry

3.8.1 Scope of Stereochemistry in SMILES

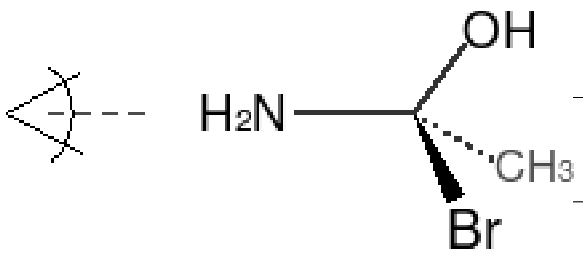
A SMILES string can specify the cis/trans configuration around a double bond, and can specify the chiral configuration of specific atoms in a molecule.

SMILES strings do **not** represent all types of stereochemistry. Examples of stereochemistry that cannot be encoded into a SMILES include:

- Gross conformational left or right handedness such as helicenes
- Mechanical interferences, such as rotatable bonds that are constrained by mechanical interferences
- Gross conformational stereochemistry such as the shape of a protein after folding

3.8.2 Tetrahedral Centers

SMILES uses an atom-centered chirality specification, in which the atom's left-to-right order in the SMILES string itself is used as the basis for the chirality marking.

Tetrahedral Chirality	
look from N towards C (chiral center)	list the neighbors anticlockwise
	`N[C@](Br)(O)C`
	. . . or clockwise
	`N[C@@](Br)(C)O`

For the structure above, starting with the nitrogen atom, one "looks" toward the chiral center. The the remaining three neighbor atoms are written by listing them in anticlockwise order using the `'@'` chiral property on the atom, or in clockwise order using the `'@@'` chiral property, as illustrated above. The `'@'` symbol is a "visual mnemonic" in that the spiral around the character goes in the anticlockwise direction, and means "anticlockwise" in the SMILES string (thus, `'@@'` can be thought of as anti-anti-clockwise).

A chiral center can be written starting from any of its neighbor atoms, and the choice of whether to list the remaining neighbor in clockwise or anticlockwise order is also arbitrary. The following SMILES are all equivalent and all specify the exact same chiral center illustrated above:

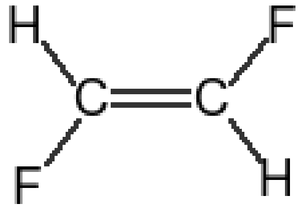
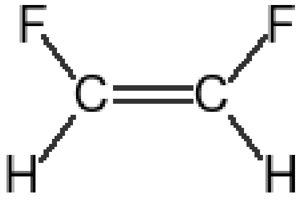
Equivalent SMILES	
`N[C@](Br)(O)C`	`Br[C@](O)(N)C`
`O[C@](Br)(C)N`	`Br[C@](C)(O)N`
`C[C@](Br)(N)O`	`Br[C@](N)(C)O`
`C[C@@](Br)(O)N`	`Br[C@@](N)(O)C`

If one of the neighbor atoms is a hydrogen and is represented as an atomic property of the chiral center (rather than explicitely as `[H]`), then it is considered to be the first atom in the clockwise or anticlockwise accounting. For example, if we replaced the bromine in the illustration above with a hydrogen atom, its SMILES would be:

Implicit Hydrogen
`N[C@H](O)C`

3.8.3 Cis/Trans configuration of Double Bonds

The configuration of atoms around double bonds is specified by the bond symbols `'/'` and `'\'`. These symbols always come in pairs, and indicate cis or trans with a visual "same side" or "opposite side" concept. That is:

Depiction	**SMILES**	**Name**
	`F/C=C/F`	trans-difluoroethane **(both SMILES are equivalent)**
	`F\C=C\F`	
	`F\C=C/F`	cis-difluoroethane **(both SMILES are equivalent)**
	`F/C=C\F`	

The "visual interpretation" of the `'/'` and `'\'` symbol is that they are thought of as bonds that "point" above or below the allenal carbon. That is, `F/C=C/Br` means "The `F` is below the first carbon, and the `Br` is above the second carbon," leading to the interpretation of a trans configuration.

This notation can be confusing when parentheses follow one of the allenal carbons:

SMILES	**Name**
`F/C=C/F`	trans-difluoroethane
`C(\F)=C/F`	
`F\C=C/F`	cis-difluoroethane
`C(/F)=C/F`	

The "visual interpretation" of the "up-ness" or "down-ness" of each single bond is **relative to the carbon atom**, not the double bond, so the sense of the symbol changes when the fluorine atom moved from the left to the right side of the allenal carbon atom.

Note: This point was not well documented in earlier SMILES specifications, and several SMILES interpreters are known to *interpret the* `'/'` *and* `'\'` *symbols incorrectly.*

A SMILES with conflicting up/down specifications is invalid:

SMILES	Comment
`C/C(\F)=C/F`	Invalid SMILES: Both the methyl and fluorine are "down" relative to the first allenal carbon

It is permissible, but not required, that every atom attached to a double bond be marked. As long as at least two neighbor atoms, one on each end of the double bond, is marked, the "up-ness" or "down-ness" of the unmarked neighbors can be deduced.

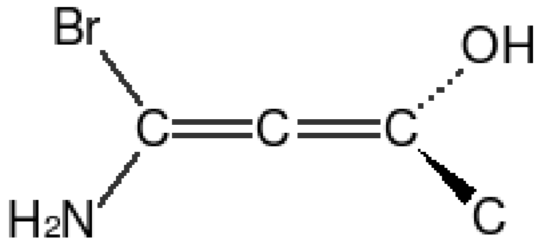
SMILES	Comment
`F/C(CC)=C/F`	trans-difluoro configuration, position of methyl is implied

Extended cis and trans configurations can be specified for conjugated allenes with an odd number of double bonds:

SMILES	Name
`F/C=C=C=C/F`	trans-difluorobutatriene
`F/C=C=C=C\F`	cis-difluorobutatriene

3.8.4 Tetrahedral Allene-like Systems

Extended tetrahedral configurations can be specified for conjugated allenes with an even number of double bonds. The normal tetrahedral rules using `'@'` and `'@@'` apply, but the "neighbor" atoms to which the chirality refers are at the ends of the allenal system. For example:

Depiction	SMILES
	`NC(Br)=[C@]=C(O)C`

To determine the correct clockwise or anticlockwise specification, the allene is conceptually "collapsed" into a single tetrahedral chiral center, and the resulting chirality is marked as a property of the center atom of the extended allene system.

3.8.5 Partial Stereochemistry

SMILES allows partial stereochemical specifications. It is permissible for some chiral centers or double bonds to have stereochemical markings in the SMILES, while others in the same SMILES string do not. For example:

SMILES	Comment
`F/C=C/C/C=C\\C`	completely specified
`F/C=C/CC=CC`	partially specified
`N1[C@H](Cl)[C@@H](Cl)C(Cl)CC1`	partially specified

3.8.6 Other Chiral Configurations

The SMILES language supports a number of atom-centered chiral configurations:

SMARTS	Configuration
TH	Tetrahedral
AL	Allenal
SP	Square Planar
TB	Trigonal Bipyramidal
OH	Octahedral

The shorthand notations '@' and '@@' correspond to anti-clockwise and clockwise tetrahedral chirality, and are the same a '@TH1' and '@TH2', respectively. Likewise, in an allenal configuration, the shorthand notations '@' and '@@' correspond to '@AL1' and '@AL2', respectively.

Very few SMILES systems actually implement the rules for SP, TB or OH chirality.

NEED COMPLETE DOCUMENTATION for SP, TB and OH.

3.9 Parsing Termination

A SMILES string is terminated by a whitespace terminator character (space, tab, newline, carriage-return), or by the end of the string.

Other data or information, such as a name, properties, registration number, etc., may follow the SMILES on a line after the whitespace character. SMILES parsers will ignore this data, although applications that use the SMILES parser will often make use of it.

3.10 Programming Practices

OpenSMILES is designed to facilitate exchange of chemical information. To achieve that goal, it SMILES parsers should impose as few limits as possible on the language.

There is no formal limit to the length of a SMILES string; SMILES of over 1 million characters have been assembled for various purposes. There is no requirement that a SMILES parser must be able to parse these exceptionally long SMILES, but as a guideline, all implementations of SMILES parsers should, at a minimum, accept and correctly parse SMILES strings of 100,000 characters. If a SMILES parser encounters a string that is too long to parse, it should generate a relevant error message.

A SMILES parser should accept at least four digits for the atom class, and the values 0 to 9999.

There is no formal limit to the number of rings a molecule can contain. There are only 100 ring-closure numbers, but since numbers can be reused, a molecule can potentially have more than 100 rings. SMILES parsers should accept and correctly parse molecules with at least 1000 rings; it is preferable to place no limits on the number of rings a molecule can contain.

Branches (parentheses) can be nested to an arbitrary depth. Some SMILES strings in standard databases contain over 30 levels of branches, and much deeper nesting is possible. A general purpose parser must handle at least 100 levels; it is preferrable to place no limits on nesting depth for parentheses.

There is no formal limit on the number of bonds an atom can have. SMILES parsers should allow at least ten bonds to each atom; it is preferrable to place no limits on the number of bonds to each atom.

There is no limit to the number of "dot-disconnected" fragments in a SMILES. A SMILES of 100,000 atoms could in principle contain no bonds at all; SMILES parsers should place no limits on the number of fragments allowed (except that it is limited to the number of atoms the parser can manage).

Programmers are **strongly** encouraged to provide detailed and clear error messages. If possible, the error message should show exactly which character or "phrase" of the SMILES string triggered the error message.

4 Writing SMILES: Normalizations

4.1 What is Normalization?

A wide variety of SMILES strings are acceptable as input. For example, all of the following represent ethanol:

SMILES	Name
`CCO`	ethanol
`OCC`	ethanol
`C(O)C`	ethanol
`[CH3][CH2][OH]`	ethanol
`[H][C]([H])([H])C([H])([H])[O][H]`	ethanol

However, it is desirable to write SMILES in more standard forms; the first two forms above are preferred by most chemists, and require fewer bytes to store on a computer. Several levels of normalization of SMILES are recommended for systems that generate SMILES strings. Although these are not mandatory in any sense, they should be considered guidelines for software engineers creating SMILES systems.

4.2 No Normalization

The simplest "normalization" is no normalization. SMILES can be written in any form whatsoever, as long as they meet the rules for SMILES. Some examples of systems that might produce un-normalized SMILES are:

- A system that enumerates combinatorial libraries using the rnum/dot-bond technique discussed above. SMILES produced by such a system will typically be a series of partial SMILES that are concatenated with dots into a complete molecule.
- Simple pass-through "filters" that don't have a full SMILES writer, but merely copy the input SMILES to the output. An example might be a molecular modeling program that reads SMILES to generates logP values, but has no capability to convert its molecular data structures back to a SMILES; instead it just copies its input SMILES to its output.

4.3 Standard Form

The "standard form" of a SMILES is designed to produce a compact SMILES, and one that is human readable (for smaller molecules).

In addition, a normalized SMILES has the important property that it matches itself as a SMARTS string. This is a very important feature of normalized SMILES in cheminformatics systems.

Note: In the example below, the "Wrong" SMILES examples are all valid SMILES, but are "wrong" in the sense that they are not the preferred form for standard normalization.

4.3.1 Atoms

Correct	Wrong	Normalization Rule
`CC`	`[CH3][CH3]`	Write atoms in the "organic subset" as bare atomic symbols whenever possible.
`[CH3-]`	`[CH3-1]`	If the charge is `+1` or `-1`, leave off the digit.
`C[13CH](C)C`	`C[13CH1](C)C`	If the hydrogen count is 1, leave off the digit.
`[CH3-]`	`[C-H3]`	Always write the atom properties in the order: Chirality, hydrogen-count, charge.
`C[C@H](Br)Cl`	`C[CH@](Br)Cl`	
`[CH3-]`	`[H][C-]([H])[H]`	Represent hydrogens as a property of the heavy atom rather than as explicit atoms, unless other rules (e.g. `[2H]`) require that the hydrogen be explicit.

4.3.2 Bonds

Correct	Wrong	Normalization Rule
`CC`	`C-C`	Only write `'-'` (single bond) when it is between two aromatic atoms. Never write the `':'` (aromatic bond) symbol. Bonds are single or aromatic by default (as appropriate).
`c1ccccc1`	`c:1:c:c:c:c:c:1`	

Correct	Wrong	Normalization Rule
c1ccccc1-c2ccccc2	c1ccccc1c2ccccc2	

4.3.3 Cycles

Correct	Wrong	Normalization Rule
c1ccccc1C2CCCC2	c1ccccc1C1CCCC1	Don’t reuse ring-closure digits.
c1ccccc1C2CCCC2	c0ccccc0C1CCCC1	Begin ring numbering with 1, not zero (or any other number)
CC1=CCCCC1	CC=1CCCCC=1	Avoid making a ring-closure on a double or triple bond. For the ring-closure digits, choose a single bond whenever possible.
C1CC2CCCCC2CC1	C12(CCCCC1)CCCCC2	Avoid starting a ring system on an atom that is in two or more rings, such that two ring-closure bonds will be on the same atom.
C1CCCCC1	C%01CCCCC%01	Use the simpler single-digit form for rnums less than 10.

4.3.4 Starting Atom and Branches

Correct	Wrong	Normalization Rule
OCc1ccccc1	c1cc(CO)ccc1	Start on a terminal atom if possible.
CC(C)CCCCCC	CC(CCCCCC)C	Try to make "side chains" short; pick the longest chains as the "main branch" of the SMILES.
OCCC	CCCO	Start on a heteroatom if possible.
CC	C1.C1	Only use dots for disconnected components.

4.3.5 Aromaticity

Correct	Wrong	Normalization Rule
c1ccccc1	C1=CC=CC=C1	Write the aromatic form in preference to the Kekule form.

4.3.6 Chirality

Correct	Wrong	Normalization Rule
BrC(Br)C	Br[C@H](Br)C	Remove chiral markings for atoms that are not chiral.
FC(F)=CF	F/C(/F)=C/F	Remove cis/trans markings for double bonds that are not cis or trans.

4.4 Canonical SMILES

A *Canonical SMILES* is one that follows the Standard Form above, and additionally, always writes the atoms and bonds of any particular molecule in the *exact same order*, regardless of the source of the molecule or its history in the computer. Here are a few examples of Canonical versus non-Canonical SMILES:

Canonical SMILES	Non-canonical	Name
OCC	CCO	ethanol
	C(C)O	
Oc1ccccc1	c1ccccc1O	phenol
	c1(O)ccccc1	
	c1(ccccc1)O	

The primary use of Canonical SMILES is in cheminformatics systems. A molecule’s structure, when expressed as a canonical SMILES, will always yield the same SMILES string, which allows a chemical database system to:

- Create a unique name (the SMILES) for each molecule in the system
- Consolidate data about one molecule from a variety of sources into a single record
- Given a molecule, find its record in the database

Canonical SMILES should *not* be considered a universal, global identifier (such as a permanent name that spans the WWW). Two systems that produces a canonical SMILES may use different rules in their code, or the same system may be improved or have bugs fixed as time passes, thus changing the SMILES it produces. A Canonical SMILES is primarily useful in a single database, or a system of related databases or information, in which all molecules were created using a single canonicalizer.

The rules (algorithms) by which the canonical ordering of the atoms in a SMILES are generated are quite complex, and beyond the scope of this document. There are many chemistry and mathematical graph-theory papers describing the canonical labeling of a graph, and writing a canonical SMILES string. See the Appendix for further information.

Those considering Canonical SMILES for a database system should also investigate InChI, a canonical naming system for chemicals that is an approved IUPAC naming convention.

4.5 SMILES Files

SMILES file consists of zero or more SMILES strings, one per line, optionally followed by at least one whitespace character (space or tab), and other data. There can be no leading whitespace before the SMILES string on a line. The optional whitespace character and data that follows it are not part of the SMILES specification, and interpretation of this data is up to applications that use the SMILES file. Each line of the file is terminated by either a singe LF character, or by a CR/LF pair of characters (commonly called the "Unix" and "Windows" line terminators, respectively). A SMILES parser must accept either line terminator. A blank line in the SMILES file, or a line that begins with a whitespace character, should be completely ignored by a SMILES parser.

5 Nonstandard Forms of SMILES

Several SMILES-generating systems are in use that either generate incorrect SMILES, or that interpreted some of the ambiguous features of the original SMILES specification in different ways. Although these SMILES are illegal according to this formal OpenSMILES specification, it is often useful to parse them, in order to make use of the information that accompanies these SMILES.

hese "relaxed" SMILES rules should only be allowed when the user (presumably after thinking about the consequences) requests it. A SMILES parser that allows any or all of these "relaxed" rules *must not do it by default*. The user must specifically request these relaxed rules before a parser can accept such SMILES.

The following table lists "relaxed" rules that SMILES parsers may accept.

Rule	**Example**	**Interpreted as ...**	**Details**
Extra parentheses	`C((C))O`	`C(C)O`	Extra parentheses are ignored in places where there is no ambiguity as to the meaning. Note that the form `(CO)N` is never allowed, since it isn't clear which atom the nitrogen should connect to.
	`C((C))O`	`C(C)O`	
	`(N1CCCC1)`	`N1CCCCC1`	
Misplaced dots	`[Na+]..[Cl-]`	`[Na+].[Cl-]`	Two or more dot-bonds in a row are condensed into one. A leading or trailing dot-bond is ignored. Note that a dot that starts a branch is *legal* in strict SMILES; for example, `C1CC(.[Na+])CC1[O-]` is a legal (though strange) SMILES.
	`.CCO`	`CCO`	
	`CCO.`	`CCO`	
Mismatched Ring Bonds	`C1CCC`	`CCCC`	Mismatched ring bonds are ignored. Note that this is almost always a bad idea. For example, `C1CCCCC2` is almost certainly supposed to be cyclohexane `C1CCCCC1`, but with "relaxed" parsing would be interpreted as hexane.

Rule	Example	Interpreted as ...	Details
Invalid Cis/Trans specification	C/C=C	CC=C	Mismatched or incomplete cis/trans bonds are ignored.
	C/C=CC	CC=CC	
	CC/=C/C	CC=CC	
Conflicting cis/trans specification	C/C(\F)=C/C	CC(F)=CC	Conflicting cis/trans bonds are ignored. (In this case, both the methyl and fluorine on the left are shown as *trans* to the methyl on the right, an impossible configuration.)
D and T	D[CH3]	[2H][CH3]	The symbols D and T are treated as synonyms for [2H] and [3H].
	T[CH3]	[3H][CH3]	
Lowercase as sp^2	CccccC	CC=CC=CC 2,4-hexadiene	Lowercase letters are interpreted as sp^2, even outside of ring systems.
	Ccc	CC=C propene	

6 Proposed Extensions

6.1 External R-Groups

Daylight proposed, and OpenEye actually implemented, an extension that specifies bonds to external R-groups. An external R-group is specified using ampersand '&' followed by a ring-closure specification (either a digit, or '%' and two digits). However, unlike ring-closures, the bond is to an external, unspecified R-group. Example: n1c(&1)c(&2)cccc1 - 2,3-substituted pyridine.

6.2 Polymers and Crystals

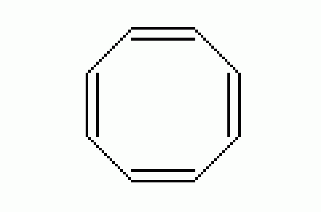
Daylight (Weininger) proposed, but never implemented, an extension for crystals and polymers. Daylight also used the ampersand '&' character, (which may conflict with the R-group proposal, above), but with the added rule that if a number appears more than once, it creates a repeating unit.

SMILES	Name
c1ccccc1C&1&1	polystyrene
C&1&1&1&1	diamond
c&1&1&1	graphite

6.3 '@' for Cis/Trans around Double Bonds

The '/' and '\' marks for cis/trans bonds seem simple on the surface but are problematic for complex systems. For example, in a long series of conjugated double bonds, changing the configuration of one bond can require rewriting dozens of bond symbols.

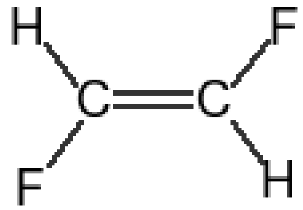
More importantly, there is a theoretical flaw with the use of '/' and '\'. In a cyclo-ene (*name??*) ring with an even number of double bonds, it is not possible to write a valid SMILES. (Recall that '/' and '\' reverse sense if moved from the left to the right of the atom, thus C/1=C/CCCCCC1 represents a cis configuration even though '/' appears twice.)

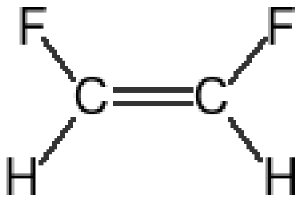
Depiction	SMILES	Comment
	C/1=C/C=C\C=C/C=C\1	Illegal SMILES

The SMILES above is illegal because the first and second C1 have opposite bond symbols '/' versus '\', so the single bond that was "broken" to create the ring has to be both "up" and "down". This logical flaw follows from using "up" and "down" in a linear notation, when in fact the atoms form a circle.

The proposed syntax for cis/trans configurations uses the `'@'` symbol on the allenal atoms. For example:

Depiction	SMILES	Name

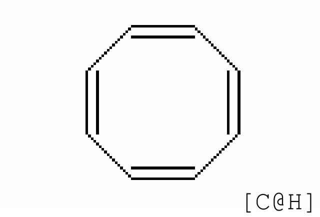
	`F[C@@H]=[C@H]F`
	`F[C@H]=[C@@H]F`
		trans-difluoroethene

	`F[C@H]=[C@H]F`
	`F[C@@H]=[C@@H]F`
		cis-difluoroethene

Interpretation of `'@'` and `'@@'` follows the tetrahedral convention: The atoms, as encountered in the SMILES string, are either in anticlockwise `'@'` or clockwise `'@@'` order as viewed on the page. Since cis/trans configurations are planar, they can also be "viewed from underneath the page", which results in the two valid SMILES shown for each compound, above.

Note that in all cases, cis and trans are easy for the chemist to distinguish visually: A trans form always has opposite "clock-ness" (`@`,`@@` or `@@`,`@`), and the cis form always has the same "clock-ness" (`@`,`@` or `@@`,`@@`) for the allenal atoms.

This proposed form of cis/trans specification using `'@'` and `'@@'` does not suffer from the theoretical flaw illustrated above:

Depiction	SMILES	Name
	`[C@H]1=[C@@H][C@@H]=[C@@H][C@@H]=[C@@H][C@@H]=[C@@H]1`	cyclooctatetraene

Note that the first allenal carbon must be represented as `'@'` since the `'1'` follows the `H`, whereas the rest of the allenal carbons use `'@@'` to characterize the cis configuration of each bond. Since this is a specification on the atom, rather than the single bond, no conflict arises at the ring-closure bond.

6.4 Radical

This section needs considerable work. The following text is courtesy Chris Morley, who commented: "I guess the last paragraph doesn't look too good in a formal specification. There are two reasons for the frailty: lack of proof that the radical and aromatic uses can always be unambigous (I doubt anybody has tried); and a known deficiency in the parser." However, it is a good starting point...

A single lowercase symbol is interpreted as a radical center. `CCc` is an alternative to `CC[CH2]` and is the 1-propyl radical; `CcC` or `C[CH]C` is the 2-propyl radical, `Co` is the methoxy radical. An odd number of adjacent lowercase symbols is a delocalised conjugated radical. So `Cccccc` is `CC=CC=C[CH2]` or `CC=C[CH]C=C` or `C[CH]C=CC=C` Lowercase `'c'` or `'n'` can be used in a ring: `C1cCCCC1` is the cyclohexyl radical.

The use of the non-aromatic lowercase symbol is a shorted form with improved intelligibilty that allows the use of implicit hydrogen in radicals. However it is intended only for simple unambiguous molecules and is not reliable when combined with aromatic atoms.

6.5 Twisted SMILES

An interesting extension that specifies conformational information via bond dihedral angles and bond lengths was proposed by McLeod and Peters:

```
http://www.daylight.com/meetings/mug03/McLeod/MUG03McLeodPeters.pdf
```

7 APPENDIX 1: References and Citations

7.1 Groups

Blue Obelisk

http://blueobelisk.sourceforge.net/

7.2 Documentation

Daylight

http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html

The most-referenced definition of SMILES on the web.

http://www.daylight.com/meetings/mug01/Sayle/m4xbondage.html

Article by Roger Sayle about converting PDB files to SMILES with thorough treatment of aromaticity.

OpenEye

http://www.eyesopen.com/docs/html/pyprog/TheSMILESLineNotation.html

http://www.eyesopen.com/docs/html/pyprog/DaylightSMILES.html

OpenBabel

http://openbabel.sourceforge.net/wiki/Radicals_and_SMILES_extensions

http://openbabel.sourceforge.net/wiki/SMILES

Wikipedia

http://en.wikipedia.org/wiki/Simplified_molecular_input_line_entry_specification

7.3 Toolkits

OpenBabel

http://openbabel.sourceforge.net/wiki/Main_Page

The Chemistry Development Kit

http://cdk.sourceforge.net/

Marvin

http://www.chemaxon.com/marvin/doc/user/smiles-doc.html

RDKit

http://www.rdkit.org/

Frowns

http://frowns.sourceforge.net/frowns.html

PerlMol.org

http://search.cpan.org/\~itub/Chemistry-File-SMILES-0.45/SMILES.pm

InChI

http://www.iupac.org/inchi/

http://inchi.info/

http://en.wikipedia.org/wiki/International_Chemical_Identifier

7.4 Some Key Scientific Papers

- David Weininger, SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules, Journal of Chemical Information and Computer Sciences, 1988, 28:31-36.
- David Weininger, Arthur Weininger, and Joseph L Weininger, SMILES 2. Algorithm for Generation of Unique SMILES Notation Journal of Chemical Information and Computer Sciences, 1989, 29:97-101.
- Morgan's original canonicalization paper: Morgan, H.L. J.Chem.Doc. 1965, 5, 107
- G.M. Downs et al, Review of Ring Perception Algorithms for Chemical Graphs, J. Chem. Inf. Comput. Cci. 1989, 29, 172-187
- R.Balducci and R, Pearlman, Novel Algorithms for the Rapid Perception of a Unique Optimal Set of Rings, J. Am. Chem. Soc. (date?)

7.5 Molecule Editors that can produce SMILES

- ChemWriter
- JME
- CACTVS
- ISISDraw?
- ChemDraw
- ACD/ChemSketch
- MarvinSketch

8 Revision History

Revision	Date	Description	Name
1.0	2007-11-13	Draft	Craig A. James
1.0	2012-09-29	Reformatting	Tim Vandermeersch
1.0	2012-09-29	Corrections	Anrew Dalke & Tim Vandermeersch

- ChangeLog
- Discussion Summary